Mobile App Development Report

13137472 Suleman Anwar

Table of Contents

Task 1 – Mobile Tracking

The first task was to develop a mobile tracking system, which will track mobile phone and upload the location details to a server, which will record these details. Users of the server system will be able see the results in a JSON strings format and also the database should store the location, userid, date/time of the location data.

This is the UploadLocation servlet where it will display the location on the server and also store it in the database.

I created four private strings, which will get the users latitude, longitude and email.

In the init method, the database drive will be loaded and the servlet will get a connection to the database with my own user and password.

```
@WebServlet("/UploadLocation")

public class UploadLocation extends HttpServlet {
      private static final long serialVersionUID = 1L;

    // This version of DisplayLocation uses a database connection
    // For efficiency, the connection is created in the init() method
    // making it available to all instances of the servlet running, as opening the connection
    // can take time, as well as closing it. This keeps an open connection.
    // Note: Investigate use of Connection Pooling to make it safer

      private String email = "unknown";
      private String latitude = "0";
      private String longitude = "0";
      private String alt = "0";
      Connection conn = null;
      Statement stmt;

      public void init(ServletConfig config) throws ServletException {
      // init method is run once at the start of the servlet loading
      // This will load the driver and establish a connection
        super.init(config);
        String user = "anwars";
        String password = "Stamwodj9";
        String url = "jdbc:mysql://mudfoot.doc.stu.mmu.ac.uk:3306/"+user;

        // Load the database driver
        try {  Class.forName("com.mysql.jdbc.Driver").newInstance();
            } catch (Exception e) {
                System.out.println(e);
            }

            // get a connection with the user/pass
            try {
                conn = DriverManager.getConnection(url, user, password);
                // System.out.println("DEBUG: Connection to database successful.");
                stmt = conn.createStatement();
            } catch (SQLException se) {
                System.out.println(se);
            }
      } // init()
```

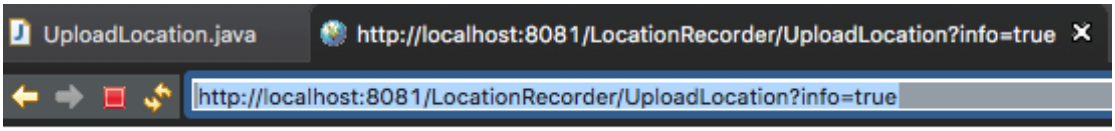
```
public void doGet(HttpServletRequest request,
          HttpServletResponse response)
throws ServletException, IOException {

    response.setStatus(HttpServletResponse.SC_OK);

    // Determine the parameters passed to the servlet
    String info = request.getParameter("info");

    if (info == null){
        // no request for info, so must be sending data
        // get data if available for each position parameter
        email = (request.getParameter("email") != null)?
                request.getParameter("email") : email;
        latitude = (request.getParameter("lat") != null)?
                request.getParameter("lat") : latitude;
        longitude = (request.getParameter("lon") != null)?
                request.getParameter("lon") : longitude;
        alt = (request.getParameter("alt") != null)?
                request.getParameter("alt") : alt;

        // DEBUG Data - print current data to server console
        System.out.println("DEBUG: Received position data. Now: Lat:"+latitude+
                           ", Lon:"+longitude+", alt:"+alt+", email:"+email);

        int numInserts = 0; // how many inserts were made
        try {
            // Create the INSERT statement from the parameters
            // set time inserted to be the current time on database server
            String updateSQL =
                "insert into LocationTrace(UserID,Latitude,Longitude,Altitude,TimeInserted)
                "values('"+email+"',"+latitude+", "+longitude+","+alt+",now());";
                System.out.println("DEBUG: Update: " + updateSQL);
                numInserts = stmt.executeUpdate(updateSQL);
        } catch (SQLException se) {
            System.out.println(se);
        }

        // send result to client as name=value pair
        PrintWriter out = response.getWriter();
        out.print("inserted="+numInserts);
        System.out.println("DEBUG: Inserted: " + numInserts);
        out.close();
    }
    else {
        // info parameter is present, so is request for last position data
        // Return current data (JSON format)
        response.setContentType("text/plain");
        String json = "{\"info\": {" +
        "\"email\": \"" + email + "\", " +
        "\"lat\": " + latitude + ", " +
        "\"lon\": " + longitude + " , " +
        "\"alt\": " + alt + "} }";
        PrintWriter out = response.getWriter();
        // System.out.println("UploadLocation JSON: "+json);
        out.print(json);
        out.close();
    }
}
```

In the doGet method, I have requested for the users email, latitude, longitude which are passed as parameters to the servlet and if there is any data then it will request it, if not then it will send the data.

It will then print out the users latitude, longitude and email to the server.

After it has sent the data to the server it will then store the data of the users latitude, longitude and email in the database using an insert statement.

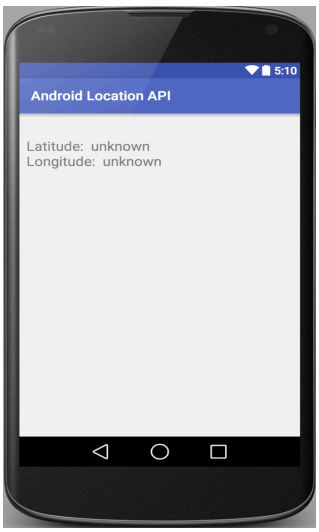
Finally, the servlet will return the data that has been saved into the database in a JSON format.



{"info": {"email": "unknown", "lat": 0, "lon": 0 , "alt": 0} }

Here is the returned data in a JSON format.

Here is the data that was saved in the database.



This is what the mobile app will output and return the data of the latitude and longitude of the user.

Task 2 – Sensor recording/actuation

In task 2, I managed to send data from two sensors to the server and also make an actuator to turn a motor and store the details in database and return the data in JSON format.

In the SensorToServer method, it will get the device type, serial number and device version of the sensors. I have also added listeners for the devices.

The method will keep looping whenever the sensor value of the sensor changes.

```
public class SensorToServer implements SensorChangeListener, InputChangeListener,
        AttachListener, DetachListener, ErrorListener, OutputChangeListener {

    public static int LED_PORT=7;
    public static int INPUT_PORT=3;
    public static int ANALOG_PORT=7;
    public static String sensorName = new String();
    public static String sensorServerURL = "http://localhost:8081/PhidgetServer/sensorToDB";
     public static void main(String[] args) throws PhidgetException {

        new SensorToServer();
    }

    public SensorToServer() throws PhidgetException {

        InterfaceKitPhidget phid = new InterfaceKitPhidget();

        try {

            phid.addAttachListener(this);
            phid.addDetachListener(this);
            phid.addSensorChangeListener(this);
            phid.addInputChangeListener(this);
            //phid.addOutputChangeListener(this);
            phid.openAny();

            phid.waitForAttachment();

            System.out.println(phid.getDeviceType());
            System.out.println("Serial Number " + phid.getSerialNumber());
            System.out.println("Device Version " + phid.getDeviceVersion());

            System.out.println("Looping...\n");
            boolean led = true;
            boolean dummy = true;
            for (; dummy;) {

                phid.setOutputState(LED_PORT, led);
                if (!phid.getInputState(INPUT_PORT))
                    led = !led;
                try {
                    // changing the output too often seems to mess up the sensor
                    // box
                    // Thread.sleep(100 + phid.getSensorValue(7));
                    Thread.sleep(1000);
                    // Thread.sleep(1000);
                } catch (Throwable t) {
                    t.printStackTrace();
                }
            }
        } finally {
            phid.close();
            System.out.println("Closed and exiting...");
        }
    }
```

```
    public void sensorChanged(SensorChangeEvent arg0) {

        // System.out.println(arg0);
        if(arg0.getIndex()==0)
        {
            sensorName= "lightSensor";
        }
            int sensorValue = arg0.getValue();
        //System.out.println("Slider value is now "+sensorValue);
        String sendResult = sendToServer(sensorValue, sensorName);
        try {
                System.out.println("Sleeping.... 1 sec");
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
       System.out.println("Sent to server, result: "+sendResult);
    }
public void inputChanged(InputChangeEvent arg0) {
        System.out.println(arg0);
    }

    public void attached(AttachEvent arg0) {
        System.out.println(arg0);
    }

    public void detached(DetachEvent arg0) {
        System.out.println(arg0);
    }

    public void error(ErrorEvent arg0) {
        System.out.println(arg0);
    }

    public void outputChanged(OutputChangeEvent arg0) {
        System.out.println(arg0);
    }
    public String sendToServer(int sensorValue, String sensorName){
        URL url;
        HttpURLConnection conn;
        BufferedReader rd;
        String fullURL = sensorServerURL + "?SensorValue="+sensorValue+"&SensorName="+sensorName+"";
       System.out.println("Sending data to: "+fullURL);
        String line;
        String result = "";
        try {
           url = new URL(fullURL);
           conn = (HttpURLConnection) url.openConnection();
           conn.setRequestMethod("GET");
           rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
           while ((line = rd.readLine()) != null) {
               result += line;
           }
           rd.close();
        } catch (Exception e) {
           e.printStackTrace();
        }
        return "done : "+result;
//        return result;

    }
}
```

In the sensorChanged method, the method will detect if there are different sensors. It will also check for the sensor value and send it to the server.

```
SensorToServer [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_75.jdk/Contents/Home/bin/java (10 Dec 2015, 23:59:46)
Sending data to: http://localhost:8081/PhidgetServer/sensorToDB?SensorValue=0&SensorName=lightSensor
Sleeping.... 1 sec
Sent to server, result: done : OK
Sending data to: http://localhost:8081/PhidgetServer/sensorToDB?SensorValue=0&SensorName=lightSensor
Sleeping.... 1 sec
Sent to server, result: done : OK
Sending data to: http://localhost:8081/PhidgetServer/sensorToDB?SensorValue=0&SensorName=lightSensor
Sleeping.... 1 sec
Sent to server, result: done : OK
Sending data to: http://localhost:8081/PhidgetServer/sensorToDB?SensorValue=63&SensorName=lightSensor
Sleeping.... 1 sec
Sent to server, result: done : OK
Sending data to: http://localhost:8081/PhidgetServer/sensorToDB?SensorValue=0&SensorName=lightSensor
Sleeping.... 1 sec
Sent to server, result: done : OK
```

Here is the sensor sending data to the server.

```java
import java.io.IOException;

/**
 * Servlet implementation class sensorToDB
 */
@WebServlet("/sensorToDB")
public class sensorToDB extends HttpServlet {

        private static final long serialVersionUID = 1L;

        private String lastValidSensorNameStr  = "no sensor";
        private String lastValidSensorValueStr = "invalid";
        private String returnMessage = "";

       Connection conn = null;
       PreparedStatement pst = null;
       ResultSet rs = null;
      Statement stmt;

      public void init(ServletConfig config) throws ServletException {
      // init method is run once at the start of the servlet loading
      // This will load the driver and establish a connection
        super.init(config);
        String user = "anwars";
        String password = "Stamwodj9";
        String url = "jdbc:mysql://mudfoot.doc.stu.mmu.ac.uk:3306/"+user;

        // Load the database driver
        try {  Class.forName("com.mysql.jdbc.Driver").newInstance();
            } catch (Exception e) {
                System.out.println(e);
            }

            // get a connection with the user/pass
            try {
                conn = DriverManager.getConnection(url, user, password);
                // System.out.println("DEBUG: Connection to database successful.");
                stmt = conn.createStatement();
            } catch (SQLException se) {
                System.out.println(se);
            }
      } // init()

      public void destroy() {
            try { conn.close(); } catch (SQLException se) {
                System.out.println(se);
            }
      } // destroy()


    public sensorToDB() {
        super();
        // TODO Auto-generated constructor stub
    }

public void doGet(HttpServletRequest request,
                  HttpServletResponse response) throws ServletException, IOException {


    response.setStatus(HttpServletResponse.SC_OK);
```

In the sensorToDB servlet, the servlet will connect to the database and store the details of the sensors from the server to the database.

In the doGet method, the servlet will get the sensor name and sensor value. It will also return no data if the sensor exists or the sensor data has not been stored.

```java
public void doGet(HttpServletRequest request,
                  HttpServletResponse response) throws ServletException, IOException {


    response.setStatus(HttpServletResponse.SC_OK);
    String info = request.getParameter("getdata");

    // Do we want info or to enter data on current sensor?
    // if request for info isn't here, record the current sensor name/value from the parameters
    if (info == null){
        String sensorNameStr = request.getParameter("SensorName");
        String sensorValueStr = request.getParameter("SensorValue");
        if (!(sensorNameStr==null) && !(sensorValueStr==null)) {
            System.out.println("DEBUG: Received sensor name: "+sensorNameStr + " with value: "+sensorValueStr);
            returnMessage = updateSensorTable(sensorNameStr, sensorValueStr);
        }
        else returnMessage = "bad data";

        PrintWriter out = response.getWriter();
        System.out.println("DEBUG: Return response for receiving data "+ returnMessage);
        out.print(returnMessage);
        out.close();

    } // endif not requesting info

else {  // send info as json
    response.setContentType("application/json");

    String sensorNameStr = request.getParameter("SensorName");
    String sensorValueStr = "No sensor data";
    if (!(sensorNameStr==null)) {
        System.out.println("DEBUG: Retrieving data for sensor name: "+sensorNameStr);
        sensorValueStr = getSensorData(sensorNameStr);
    }
    else {
        // give indication of no sensor name
        sensorNameStr = "Unknown sensor";
    }

    String json = "[{\"" + sensorNameStr +
              "\": \"" + sensorValueStr + "\"}]";

//    String json = "{\"sensor\": {\"" + lastValidSensorNameStr +
//              "\": \"" + lastValidSensorValueStr + "\"}}";

    PrintWriter out = response.getWriter();
    System.out.println("DEBUG: json return: "+json);
    out.print(json);
    out.close();
}
}
```

```
DEBUG: Update: insert into sensorUsage(SensorName, SensorValue, TimeInserted) values('lightSensor','0', now());
DEBUG: Return response for receiving data OK
DEBUG: Received sensor name: lightSensor with value: 0
DEBUG: Update: insert into sensorUsage(SensorName, SensorValue, TimeInserted) values('lightSensor','0', now());
DEBUG: Return response for receiving data OK
DEBUG: Received sensor name: lightSensor with value: 63
DEBUG: Update: insert into sensorUsage(SensorName, SensorValue, TimeInserted) values('lightSensor','63', now());
DEBUG: Return response for receiving data OK
DEBUG: Received sensor name: lightSensor with value: 0
DEBUG: Update: insert into sensorUsage(SensorName, SensorValue, TimeInserted) values('lightSensor','0', now());
DEBUG: Return response for receiving data OK
DEBUG: Return response for receiving data bad data
DEBUG: Received sensor name: lightSensor with value: 15
DEBUG: Update: insert into sensorUsage(SensorName, SensorValue, TimeInserted) values('lightSensor','15', now());
DEBUG: Return response for receiving data OK
```

Mobile_Application_Development — mysql -h mudfoot.doc.stu.mmu.ac.uk -u a...

NULL	slider	0	2015-12-08 16:03:27
NULL	slider	18	2015-12-08 16:03:28
NULL	slider	8	2015-12-08 16:03:33
NULL	slider	18	2015-12-08 16:03:38
NULL	slider	8	2015-12-08 16:07:15
NULL	slider	18	2015-12-08 16:07:26
NULL	slider	7	2015-12-08 16:11:26
NULL	slider	17	2015-12-08 16:11:31
NULL	slider	0	2015-12-08 16:11:56
NULL	slider	0	2015-12-08 16:11:58
NULL	slider	0	2015-12-08 16:11:59
NULL	slider	0	2015-12-08 16:12:00
NULL	slider	0	2015-12-08 16:12:01
NULL	slider	0	2015-12-08 16:12:02
NULL	slider	0	2015-12-08 16:12:03
NULL	slider	16	2015-12-08 16:12:04
NULL	lightSensor	0	2015-12-08 17:11:22
NULL	lightSensor	0	2015-12-08 17:11:23
NULL	lightSensor	0	2015-12-08 17:11:24
NULL	lightSensor	0	2015-12-08 17:11:25
NULL	lightSensor	0	2015-12-08 17:11:26
NULL	lightSensor	0	2015-12-08 17:11:27
NULL	lightSensor	0	2015-12-08 17:11:28
NULL	lightSensor	15	2015-12-08 17:11:29

Here is the data being saved into the database.

```
public class MotorFromServer {

    /*
     * Demo phidget code to allow a call to poll a server and request value.
     * Returned value is used to turn motor to the required angle
     * No validation is done. Value is straightforward text
     */


    // time to wait before asking server for another value
    public static int secondsToPoll = 2;
    int servoNumber = 0;
    public static String sensorServerURL = "http://localhost:8081/PhidgetServer/sensorToDB";
    public static int receivedMotorValue, lastMotorValue;
    static  AdvancedServoPhidget servo;
    public static String sensorValue;

    public static final void main(String args[]) throws Exception {
        System.out.println(Phidget.getLibraryVersion());
        servo = new AdvancedServoPhidget();
        setServoListeners();
    // main code starts here
        new MotorFromServer();
    }

    public MotorFromServer() throws PhidgetException {


        lastMotorValue=0;
        int sleepTime = 1000*secondsToPoll;
        boolean loopForever = true;
        for (; loopForever;) {
            receivedMotorValue = getMotorValue();
            System.out.println("DEBUG: Old motor value "+lastMotorValue+" new motor value "+receivedMotorValue);
            //If statement, if light is less than 500 then turn motor to 90.
            //If light is more than 500 then turn motor to 180.
            if (receivedMotorValue != lastMotorValue) {
                // move motor
                if(receivedMotorValue<500)
                {
                moveServoTo(90);
                }
                else{
                    moveServoTo(180);
                }
                lastMotorValue=receivedMotorValue;
                System.out.println("DEBUG: Moved motor to "+receivedMotorValue);
            }
            // sleep for a while
            try {Thread.sleep(sleepTime);
            } catch (Throwable t) {t.printStackTrace();
            }
        }

    }

    private int getMotorValue() {
        URL url;
        HttpURLConnection conn;
        BufferedReader rd;
        String fullURL = sensorServerURL + "?getdata=true"+"&SensorName=lightSensor";
       System.out.println("DEBUG: Requesting motor data via: "+fullURL);
        String line;
        String result = "";
        try {
           url = new URL(fullURL);
           conn = (HttpURLConnection) url.openConnection();
           conn.setRequestMethod("GET");
           rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
           while ((line = rd.readLine()) != null) {
              result += line;
           }
           rd.close();
        } catch (Exception e) {
           e.printStackTrace();
        }
        System.out.println("DEBUG: received from server value>"+result+"<");

        //Result is stored in JSONArray, extract the object which is named 'light sensor'.
        try {
            JSONArray jsonArr = new JSONArray(result);
            for(int i = 0; i<jsonArr.length(); i++)
            {
                JSONObject jsonObj = new JSONObject();
                jsonObj = jsonArr.getJSONObject(i);
                sensorValue = jsonObj.getString("lightSensor");
            }
```

In the MotorFromServer method, the servlet will poll the server and request values of the sensor and when the value has been returned, it will then turn the motor to a required angle. The method will also request the current value of the motor. I have inserted an if statement for the light sensor which will test if the light is less than 500 then turn the motor to 90 degrees, if the light is more than 500 then turn the motor to 180 degrees.

In the getMotorValue method, the method will require the motor data from the URL, which is the sensotToDB servlet so it will request data from the database and when the sensor value has been received from the database then it will return the values in a JSON format.

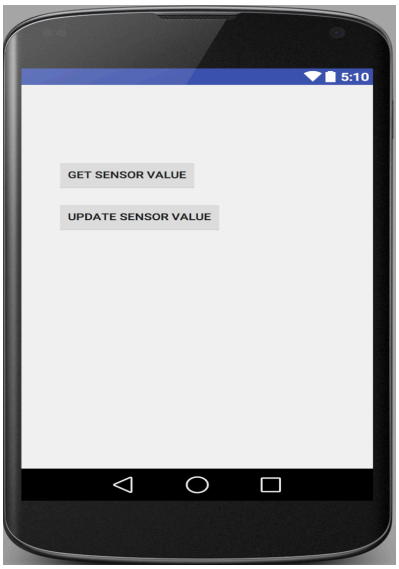
```
Phidget21 - Version 2.1.8 - Built Oct  9 2015 09:52:41
DEBUG: Requesting motor data via: http://localhost:8081/PhidgetServer/sensorToDB?getdata=true&SensorName=lightSensor
DEBUG: received from server value>[{"lightSensor": "15"}]<
SensorValue15
DEBUG: Conversion to numeric OK
DEBUG: Old motor value 0 new motor value 15
attachment of PhidgetAdvancedServo v101 #305212 (attached)
DEBUG: Moved motor to 15
```

Here is the sensor value being requested from the server and also moving the motor to a required angle.

Task 3 – Mobile Phone App Integration

In task 3, I have managed to display the sensor values via the mobile application.

I have created two buttons, which will get the sensor value and update the sensor value.




```java
public class ShowSensorData extends ActionBarActivity {
    private static final String TAG="MainActivitySensorEg";
    public static String sensorServerURL = "http://10.0.2.2:8081/PhidgetServer/sensorToDB";
    public static String sensorToGet = "motor";
    public static String sensorUpdatedValue = "90";

    public TextView sensorValueField;
    public TextView sensorDescriptorField;

    private Button mUpdateSensorValueButton;
    private Button mRetrieveSensorValueButton;

    AsyncHttpClient sensorServerClient = new AsyncHttpClient();
    //SensorServerClient sensorServerClient = new SensorServerClient();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_show_sensor_data);

        sensorValueField = (TextView) findViewById(R.id.sensorvalueTV);
        sensorDescriptorField = (TextView) findViewById(R.id.sensornameTV) ;
        mRetrieveSensorValueButton = (Button) findViewById(R.id.getSensorValueButton);
        mUpdateSensorValueButton = (Button) findViewById(R.id.updateSensorValueButton);

        mRetrieveSensorValueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startGettingData();


            }
        });

        mUpdateSensorValueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startPuttingData();
            }
        });

    }
```

```java
    private void startGettingData() {
        RequestParams params = new RequestParams();
        params.put("getdata", "true");
        params.put("sensorname", sensorToGet);
        sensorServerClient.get(sensorServerURL, params, new AsyncHttpResponseHandler() {
            @Override
            public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {
                String response = new String(responseBody);
                sensorValueField.setText(response);
            }
            @Override
            public void onFailure(int statusCode, Header[] headers, byte[] responseBody, Throwable error) {
                error.printStackTrace(System.out);
            }
        });
        Log.i(TAG, "started thread to get sensor data");
    }

    private void updateSensorDisplay(){

    }

    private void startPuttingData() {
        RequestParams params = new RequestParams();
        params.put("sensorvalue", sensorUpdatedValue);
        params.put("sensorname", sensorToGet);
        sensorServerClient.get(sensorServerURL, params, new AsyncHttpResponseHandler() {
            @Override
            public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {
                String response = new String(responseBody);
                sensorValueField.setText(response);
            }
            @Override
            public void onFailure(int statusCode, Header[] headers, byte[] responseBody, Throwable error) {
                error.printStackTrace(System.out);
            }
        });
        Log.i(TAG, "started thread to put sensor data");
    }
```

This is the ShowSensorData java file where I have used the AsyncTask to thread the sensor values from the database via the URL, which will provide the sensor values from the database.

Task 4 – Internet of Things (IoT) Analysis

Towards a Smarter and Safer Future City Environment

Background and summary of work

Over the years as technology evolves, data can be easily accessed and shared through Internet access via mobile devices especially smart phones. It has evolved so much that the users have the ability to create and share information regarding current events. Users don't need to wake up early to go to the local newsagents and buy newspapers or watch the news to keep up to date on what's happening around the world. In the next 50 years, urban population will be doubled and therefore cities will need to become smarter to reduce expenses and improve quality of life and to do that we need smart cities.

Abbas Rajabifard talks about how technology especially in mobile devices have increased over the years and by easily accessing and sharing data, can make cities smarter. The conference also talks about the challenges and opportunities faced in journey towards a smarter and safer cities. [2]

Analysis of the proposed ideas

Abbas Rajabifard idea is to use open data, big data and sensor networks to support the development of safe and smarter environments. [2]

Data

ICT is rapidly becoming universal to city environments and delivering the essential foundation for maintainability and flexibility of the smart future cities. With the rapid growth in the existence of Internet of Things (IoT), the future of the Internet and technologies in smart cities, big data is produced which will need to be managed correctly and examined for a variety of applications which will be using organised and integrated ICT approach. A smart city uses information communication technologies to improve quality, interactivity and performance of urban services to reduce costs and resource consumption to improve contact between citizens and government. Many tools for a smart city will have to deal with different application domains transport, energy and land use and it will also hardly deliver an integrated information viewpoint to deal with the maintainability and social and economic growth of the city. Smart cities can benefit from using information such as open data, big data, real-time data, sharing, processing and integration through a web service, which can inter-operate in a cloud environment. [3]

Advantage and shortcoming of data

Using such data has its advantages but also has its shortcomings. Utilising some information will require suitable software tools, technologies, services to visualise, collect, analyse and store huge amounts of data from the urban environment. The way data is gained is by performing data analytics by using various data statistical methods, mining and machine learning. Nonetheless, the field of city data will be increasingly changing, broad and complex. The city data analytics will be very complex due to a variation of issues:

- Requirements of cross-thematic applications such as water, energy, transport etc.
- Numerous sources of structured, unstructured and semi-structured data
- Reliability of data [3]

Even in private sectors, analysis of big data is too large and complex to be examined without software but by using techniques such as user profiling and personalised services can be used to improve sales. This can be used to gain real-time insights of the public's health and could focus their large on weak groups. It would offer new opportunities but also cause huge challenges such as potential abuse of data, privacy and public trust. However, the increasing use of devices that are Internet enabled with sensors will provide more opportunities in both to improve the services that are being delivered and also to connect the data to attain swift insights into whether the involvements are functioning. Not only sensors can empower citizens but mobile devices can for example tell the citizens which car park spot is available or tell a farmer when to plant the crops. [4]

Networks

Another way smart city can have a safer environment is using sensors. This will be able to change both strength and coverage of connection in almost every aspect of infrastructure. From energy networks to water and waste to transit, from the street trees to housing will be able to wirelessly transmission their activity and state in real-time through the use of low-cost, discreet and robust sensors. This concept is called Internet of Things (IoT) in which every object has some sort of network connectivity, which allows it to send and receive data. For example, the sensors could be placed on infrastructures that already exist such as installations, which will monitor mobile data, water and air quality that will show patterns of movement of people in the city. [5]

Advantage and shortcoming of networks

The advantage of using networks towards a smarter city and a safer future city environment is user experience. The citizens will sense that the city has smart interfaces, which will enable a more personalised, rich and more efficient experience. The advantage for the city will be more efficient distribution of services and use of the infrastructure and first-time strategic information, which will be used for the city and its services. With these two combining, it will show the potential of having a smart city. The more effective the delivery of the trendy services equipped with an enormously improved user experiences for the citizens, it will deliver detailed data on the services operation. The better the user experience, it will generate an increase in uptake which generate better data and better data will generate better user experience. [5]

Conclusion

To conclude, in order for smart city to have a safe environment, the government will have to access to big and open data, which may cause a huge issue for the citizens because of privacy and misuse of data. However, it can improve emergency services response time and also update the conditions of the city to the citizens. For example, if there is a group of people who are causing havoc and someone takes a video of them and puts it on social media then the police services can see it and track the group down.

References

1. https://en.wikipedia.org/wiki/Smart_city

2. IEEE ISSNIP 2015 and RIoT 2015 - Conference Proceedings

3. http://www.journalofcloudcomputing.com/content/pdf/s13677-015-0026-8.pdf

4. http://www.undatarevolution.org/data-innovation/

5. Arup_SmartCities_June2011

Turnitin originality Report